**System Design Document**

**For**

**EcoCAR UI team**

Team member:

|  |  |
| --- | --- |
| Version/Author | Date |
| 1.0/EcoCAR UI Team | 10/1/20 |
| 2.0/EcoCAR UI Team | 10/26/2020 |
|  |  |
|  |  |

TABLE OF cONTENT

[1 INTRODUCTION 3](#_Toc530500649)

[1.1 Purpose and Scope 3](#_Toc530500650)

[1.2 Project Executive Summary 3](#_Toc530500651)

[1.2.1 System Overview 3](#_Toc530500652)

[1.2.2 Design Constraints 3](#_Toc530500653)

[1.2.3 Future Contingencies 3](#_Toc530500654)

[1.3 Document Organization 3](#_Toc530500655)

[1.4 Project References 4](#_Toc530500656)

[1.5 Glossary 4](#_Toc530500657)

[2 SYSTEM ARCHITECTURE 4](#_Toc530500658)

[2.1 System Hardware Architecture 4](#_Toc530500659)

[2.2 System Software Architecture 4](#_Toc530500660)

[2.3 Internal Communications Architecture 4](#_Toc530500661)

[3 HUMAN-MACHINE INTERFACE 4](#_Toc530500662)

[3.1 Inputs 5](#_Toc530500663)

[3.2 Outputs 5](#_Toc530500664)

[4 DETAILED DESIGN 5](#_Toc530500665)

[4.1 Hardware Detailed Design 6](#_Toc530500666)

[4.2 Software Detailed Design 6](#_Toc530500667)

[4.3 Internal Communications Detailed Design 7](#_Toc530500668)

[5 EXTERNAL INTERFACES 7](#_Toc530500669)

[5.1 Interface Architecture 7](#_Toc530500670)

[5.2 Interface Detailed Design 8](#_Toc530500671)

[6 SYSTEM INTEGRITY CONTROLS 8](#_Toc530500672)

System Design Document

*Overview*

*The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.*

# INTRODUCTION

## Purpose and Scope

The purpose of this project is to provide a Human-Machine Interface (HMI) for EcoCAR that is reliable and easier to use for the user according to the external conditions.

## Project Executive Summary

The purpose of the Eco-car UI is to provide an easy to use interface for users of EcoCAR to allow them to understand and adapt to the limitations and current state of the Adaptive Cruise Control system.

This system will do this by providing a “birds-eye view’ of the nearby cars that EcoCAR has detected, so that the user is aware of what the car can and cannot see.

### System Overview

The System will consist of a 6.2 by 3.06-inch screen on the dashboard that will display information, combined with the reflective mirror that will allow the screen to be seen by the driver. This screen will display DPI information, along with relative car locations to the user.

### Design Constraints

The system needs to be easy to use for new users who are unfamiliar with the system as this system is a ridesharing service and frequently has new users who will likely not be able to read the manual before the drive or be extremely comfortable with the features.

## Document Organization

This document is organized in a manner that highlights components in the following order: system architecture, design, external interface.

## Project References

ROS Data taken from EcoCAR GitHub directory

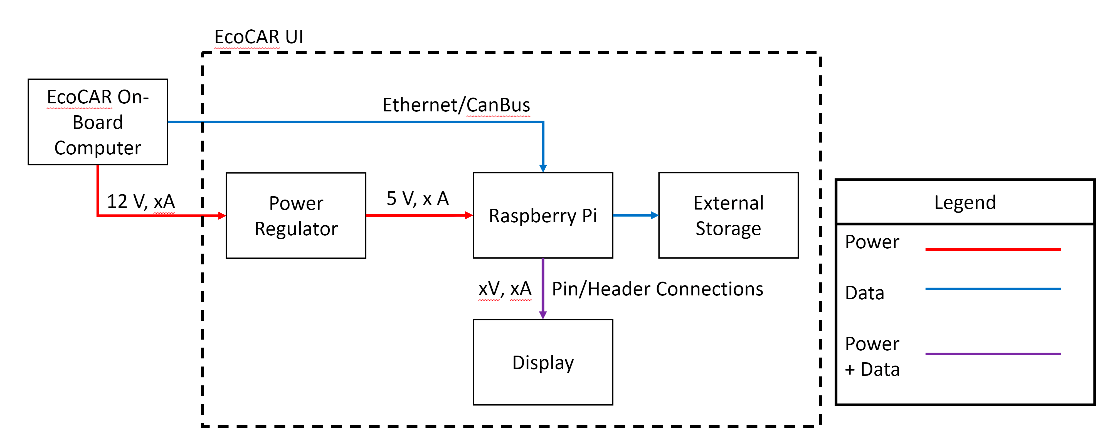
## Glossary

* ROS: Robot Operating System
* UI: User Interface
* ACC: Adaptive Cruise Control

# SYSTEM ARCHITECTURE

## System Hardware Architecture

The hardware architecture for this project is to use a Raspberry Pi that has a Touch Screen on top of it for User interface. This raspberry Pi needs to be connected to the Eco-Car core computer through either an ethernet cable or WIFI connection. It also will have a 15V to 5V Buck Converter to efficiently drop the voltage from the Car’s voltage to the Raspberry Pi’s voltage.



## System Software Architecture

The Software for this system will connect to the car over a network connection, Ethernet, in order to communicate with the core computer of the car. This will allow the data from the Adaptive Cruise Control to be passed from the core car to the EcoCAR UI subsystem. The information will be passed through an application called ROS to

This information will then be processed and displayed on a screen that will show the User some current information including Relative Car Locations as determined by the EcoCAR.

## Internal Communications Architecture

The EcoCAR will be connected to the car through a network connection, either Ethernet or WIFI, this will allow messages from ROS, specified in the various modules of ROS, to be passed from EcoCAR to the UI system. Refer to the System Hardware Architecture diagram in Section 2.1.

# HUMAN-MACHINE INTERFACE

This project will be an interface between the EcoCAR and the Driver of the car. This project will utilize the computing system in the EcoCAR to communicate the features of adaptive cruise control. The user interface/computing system will allow the driver to see relevant objects such as car, pedestrians, and obstacles in the EcoCAR. The user interface system will show the user some current information including Relative Car Locations as determined by the EcoCAR.

## Inputs

Inputs include:

* The ROS data from the car’s computer. This data will be used to get the current status of the EcoCAR.
* The driver’s inputs that control how the ACC will work.

## Outputs

The system will output the relative location of the cars through car icons that will be displayed from an overhead view of the relative locations of the various cars. In the center of the screen, a static car will be displayed representing the Eco-Car. All cars that are detected within two car lengths and within 1 lane in either direction will be displayed on the screen.

This will allow the user to clearly see the cars that are around Eco-Car

The system will also fade the top and bottom of the screen red to alert the user when they are getting too close to the car in front of them

# DETAILED DESIGN

## Hardware Detailed Design

The Raspberry Pi will need a power source of 5V 3A maximum. This can be achieved from using the car’s battery as a source and a step-down converter.

There is the possibility of either WIFI or Ethernet for communications, the Raspberry Pi has both connections natively.

The Raspberry Pi will have a SD card for storage, a 16GB or greater will be a viable option that will not need to be replaced in the future for storage constraints.

The Raspberry Pi has a 1.5GHz quad core processor based on ARM

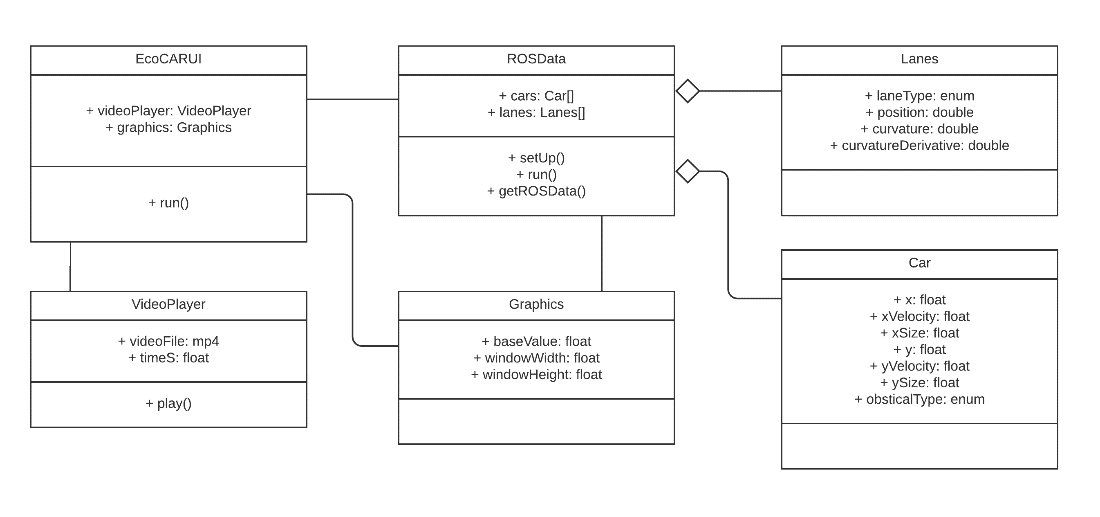
The Raspberry Pi will utilize the Raspberry Pi touchscreen

With the usage of 8in long 18awg cables the display can be moved to the top of the dash while keeping the Raspberry Pi, step down converter, and the main communications to the car under the dash.

The step-down converter can be spliced into the **Always ON** or **Accessory ON** power going to the radio/screen in the center, which will allow around a foot maximum of 14awg wire to connect to the Raspberry Pi (with it in a position where it can have the display on the dash with 8in wires to connect to that)

The Raspberry Pi touchscreen has a standard resolution of 800x480

## Software Detailed Design



The current software design has a EcoCAR UI master class which will control what the screen is by having a state and only running certain modules at a time

The Video Player Module is run when the EcoCAR has a new member and the car is turned on. This allows a basic training of new people on how the system works.

The Car Displayer module is the module that will display nearby cars on the screen for the driver to be aware of what EcoCAR can currently see and what it cannot see. This module needs to subscribe to appropriate ROS messages in order to determine the location of the cars

The Car Displayer uses the Car class as a container for the location of all the nearby cars

The Proximity Alarm is a second module that will look through EcoCAR messages except it will do so in order to alert the user if they are too close to the car in front of them, it will do this by adding a red tint to the top and bottom of the screen

## Internal Communications Detailed Design

The communication between the UI system and the EcoCAR will be through ROS messages over ethernet. These messages are specified in ROS messages in the various modules

# EXTERNAL INTERFACE

## Interface Architecture

The overall interface architecture will work on the idea of utilizing ROS (Robot Operating system) to communicate by sending and user receiving messages. The idea would be that the video will be displayed as an input when the user starts the car. This will help in the user navigating on how to use a car’s adaptive cruise control system.

The Raspberry Pi will also have a stepdown converter in order to convert the 12V of the car to the 5V of the Raspberry Pi

## Interface Detailed Design

The format of the data displayed such as the cars/objects or relative position of the car’ should be in JPEG’s. The images of the cars being displayed should be in JPEG’s. The relative position of the car should be displayed in meters.

# SYSTEM INTEGRITY CONTROLS

The user interface will start working only when the EcoCAR’s engine is turned on. All communications such as the relative positions of the shall be should be communicated with at least 98% accuracy and within 100 ms of the information being processed. This project does not have much of a concern with unauthorized access of the data. This is since the user interface is only accessible to the driver in the car only. There isn’t a need to save the data anywhere for future usage. The user interface will display the present data in the car. The present data will include relative positions of the car in respect to its traffic surroundings. Calculations will be made and displayed in the EcoCAR such as how much distance another car is relative to the EcoCAR.